Developing Adaptive Applications: The MOST Experience

A. Friday, N. Davies, G.S. Blair and K.W.J. Cheverst

Distributed Multimedia Research Group,
Department of Computing,
Lancaster University,
Lancaster,
U.K.
Fax: +44 (0)1524 593608
Telephone: +44 (0)1524 65201
E-mail: adrian, nigel, gordon, kc@comp.lancs.ac.uk

Abstract

Future computer environments will include mobile computers which will either be disconnected, weakly inter-connected by low speed wireless networks such as GSM, or fully inter-connected by high speed networks ranging from Ethernet to ATM. Such environments place unique demands on systems, requiring software to dynamically adapt to rapid and significant fluctuations in communications quality-of-service (QoS). This paper reviews existing adaptation techniques and describes an experiment in developing an adaptive mobile application and associated distributed systems platform. The experiences gained during this experiment are presented and analysed to provide a basis for the engineering of future adaptive systems.

1. Introduction

Future computer environments will include mobile computers which will either be disconnected, weakly inter-connected by low speed wireless networks such as GSM, or fully

inter-connected by high speed networks ranging from Ethernet to ATM [Davies,96]. While the transition between networks is currently a heavyweight operation we believe that developments in network interface technology will soon enable mobile computers to dynamically select their network service based on cost and performance requirements. Such flexibility, coupled with the inherent unreliability of mobile communications, means that system services and applications will be subject to rapid and massive fluctuations in the quality-of-service provided by their underlying communications infrastructure. In this paper we discuss the design of distributed systems (platforms and applications) which are able to tolerate this environment by dynamically *adapting* to changes in the available communications quality-of-service [Davies,94], [Katz,94].

Section 2 presents the case for adaptation in modern heterogeneous mobile environments. Section 3 then classifies and reviews current adaptation techniques. The classification is based on the level of the system architecture at which adaptation occurs (system, platform, application or user) and we argue that while current systems tend to support adaptation at a subset of these levels future systems will be required to adapt at all of these levels. Section 4 describes in detail the results of an experiment we have conducted into developing such a system. This work was carried out as part of the MOST project and consisted of the development of a collaborative mobile application and associated systems support which included support for adaptation at all levels. Section 5 analyses the results of our experiment and provides guidelines for designers of future adaptive systems. Finally, section 6 contains some concluding remarks and a description of our future work in this area.

2. The Requirement for Adaptation

2.1 Heterogeneous Networks

Conventional fixed hosts utilise a single networking technology to provide connectivity. In contrast, mobile hosts may have to make use of a diverse range of networks to provide connectivity during a typical operational cycle. For example, in a modern office or research environment a mobile host may be connected to a fixed network while the user remains within their home environment. As the user moves within the local environment a range of

infra-red or wireless local area networks may be used to maintain connectivity. Movement outside of the local environment may require use of a metropolitan or wide area wireless network such as Metricom or GSM (subject to availability).

The use of a range of networking technologies has a profound impact on the mobile host due to the unique character or QoS signature of each of the networks. In more detail, fixed networks typically offer reliable high bandwidth communications for minimal (perceived) cost while wireless networks offer relatively low bandwidth unreliable communications with, in the case of wide area public networks, a significant cost to the end-user. Table I presents QoS signatures for a range of common networks available to mobile hosts.

The impact of different QoS signatures on system software has been clearly demonstrated [Katz,94], [Cáceres,94], [Friday,96]. For example, while local area wireless networks such as WaveLAN appear to offer similar characteristics to their fixed counterparts they differ in a number of important aspects including increased bit error rates and control signalling leading to higher levels of packet loss. While it might naively be assumed that these packet losses can simply be treated as a reduction in the overall bandwidth available, Cácares demonstrated in [Cáceres,94] that it is important that network protocols are tuned for networks with these packet-loss characteristics. In more detail, he showed that the performance of TCP was significantly reduced when it was operated over a wireless network with packet-losses due to bit-errors and cell-handoffs. Specifically, the loss of packets causes a marked degradation in TCP's performance due to TCP's exponential back-off and slow start strategy originally developed to avoid network congestion.

The necessity for system software to be tailored for specific networking technologies together with the use of multiple networking technologies by mobile hosts implies that future system software must be able to dynamically reconfigure, or *adapt*, to match the QoS signature of the underlying communications infrastructure. The problem of supporting this requirement is further compounded by two factors: the development of technologies such as MINT [Hager,93] which allow mobile hosts to rapidly switch between overlapping networking technologies and the ability of mobile hosts to drive multiple networks concurrently [Katz,96a].

2.2 Other Factors

In addition to variation in network characteristics as identified in the previous section, mobile hosts are subject to changes in a number of other environmental factors. Notable examples of these factors include power availability and consumption, the host's physical location and the availability and location of services.

Mobile computers are always subject to the limitations imposed by current battery technology. To address this problem they use a variety of mechanisms for reducing their overall power consumption including low-power processors and i/o devices, the use of doze-modes and hardware suspend-resume capabilities and automatically reducing the processors clock-speed during periods of relative inactivity. The work of Badrinath et al. on the design of low-power distributed algorithms [Imielinski,94] has shown that with the use of software controllable doze-modes a reduction in power usage can be made by, for example, sending large computations to remote sites and dozing until the results are ready to be collected. Clearly a number of factors such as the other activities running on the mobile machine and the relative costs of transmission will determine the practical benefits of such an approach. However, mechanisms such as these highlight the requirement for mobile end-systems to adopt a flexible approach to power management.

In addition to power availability, mobile hosts must also be able to adapt to physical changes in locality. These changes can be detected in a number of ways: in the case of cellular systems it is possible to query the system to determine the position of any given user. Technologies such as the active-badge system designed at Olivetti Research Labs [Want,92] can also be used to determine the physical location of mobile users. The implication of this is that at any point in time it is possible to accurately pinpoint a user's physical location and, perhaps more importantly, the location of other services and users with which they may wish to interact. Information regarding changes in a mobile host's location can be exploited in a number of ways. Early work by [Neuman,93] on the Prospero system put forward the idea of matching client's service requests to appropriate services based on location. So, for example, asking to be supplied with a printer service and specifying a constraint such as 'location =

nearest' ensures that a client is able to print their document to the printer closest to their physical location. Location information, and the concept of the *proximate selection* of services, is also focussed upon in the work of Schilit [Schilit,94] who provides a framework for what are termed context-aware applications. An example of the types of application Schilit proposes is a memory jogger which allows users to specify that a certain message is displayed when a series of location and temporal based conditions are met, e.g. "remind me to do task t next time I meet users x and y". Finally, changes in location information can be used to explain anomalies in network transmission as in the work of Cácares.

3. Adaptation Techniques

3.1 Overview of Techniques

Section 2 has presented the case for adaptation in modern heterogeneous systems. In this section we focus on adaptation techniques for managing changes in network QoS. Table II, which summarises the principle adaptation techniques available, clearly demonstrates that adaptation is possible at every level of a system. Indeed, it is the authors' opinion that to operate effectively in a mobile environment systems will require support for adaptation to be implemented at all levels.

3.2 System Level Adaptation

The operating system, and particularly the protocol stack, is a natural place for implementing adaptive techniques. At the simplest level the operating system may support a range of protocols each tailored for a specific communications technology. When the underlying network changes the system is able to dynamically swap to a new protocol. The granularity of reconfiguration can range from systems which treat entire protocols as an atomic unit [Renesse,96] to those which are able to replace individual protocol units to reflect the changing requirements [Crane,96].

An important example of protocol level adaptation occurs when alternative protocols or protocol components are used to match the scheduling of packets to the bearer technology. In high-speed networks packet scheduling is primarily concerned with avoiding network congestion and maintaining any relevant QoS parameters associated with continuous media

transmission. In the case of dial-up network services, traffic can be scheduled using batching techniques to reduce the latency and cost of connection establishment (furthermore, in a pay per unit time system, additional data and meta data can be scheduled during idle periods up until the next charge boundary) [Davies,96]. Finally, in systems where bandwidth is extremely limited (typically those using wide area wireless communications), it may be necessary to reorder data according to some notion of priority, urgency or deadline [Joseph,96]. In the LittleWork project for instance, the SLIP daemon was modified to support additional levels of priority, in order that interactive user traffic was not caught behind less significant file system updates [Waldspurger,94].

The problems of protocol-level adaptation are complicated in instances where multiple bearers are available simultaneously. The BARWAN [Katz,96b] and Monarch [Hills,96] research testbeds offer an environment such as that described in section 2.1, consisting over multiple 'overlaid' networks. In addition to the question of how to maximise the usage of the range of available networks, it is necessary to decide which of the available alternatives are most appropriate for a given item of data. This may be determined at the protocol level (or below) as in [Hager,93] or may be based on application requirements [Katz,96a]. The issue of application level adaptation and requirements is discussed in section 3.4.

3.3 Adaptation in Middleware

A number of projects have investigated the development of services to support adaptation at the middleware (platform) level [Friday,95], [Schill,95] and three basic approaches have emerged. Firstly, the middleware services can attempt to reduce the bandwidth requirements of the application by, for example, increasing the level of compression that is applied to data before transmission. This technique is particularly effective when applied to continuous media when lossy compression techniques [Wallace,91] or filtering out of non-essential data [Zenel,95] can be used.

The second approach is to develop services which can pre-fetch and cache data during periods of high-connectivity in preparation for future reductions in network bandwidth. A refinement of this approach is to re-order data in order to pre-fetch important information

during periods of weak-connectivity. This technique is frequently used in off-line mail and news readers that prefetch summary information (such as headers and subject lines) before pulling the remainder of the article when it is needed. Supporting services can apply this technique more generally by, for example, supplying the first page of a document to an editor and then fetching successive pages while the user reads the first page.

The third approach is to dynamically rebind clients to services as network availability changes. In particular, during periods of disconnection clients can rebind to local proxy services until network connectivity can be restored. Since most platforms are based on RPC style interprocess communication paradigms which cannot tolerate interruptions to connectivity considerable work has been carried out on modifying the traditional semantics of RPC to work in a less synchronous mode. The ROVER toolkit [Joseph,95] provides the application programmer with a mechanism called Queued-RPC (QRPC) which aims to couple the API of RPC with network fault tolerance. QRPC logs RPCs to stable storage while the network link is down and replays the logged operations upon reestablishment. This type of modified RPC mechanism is more suitable as a communications paradigm in mobile environments than a traditional RPC.

Combinations of these approaches are possible: systems such as the CODA file system [Kistler,92] use pre-fetching to prime a client cache, dynamic rebinding to a local server emulation during periods of disconnection and, in recent incarnations [Mummert,94], filtering and compression to reduce the size of the replay log on reconnection.

3.3 Application Level Adaptation

As discussed in the previous section, support services can act on behalf of an application to attempt to mollify the implications of changes in the underlying communications infrastructure. In effect, the system services are attempting to provide mobility transparency. However, in many cases it is the applications themselves which best understand how to react to changes in the network QoS.

The simplest adaptation technique which applications can employ is to adjust their communications demands to meet the changes in underlying communications capabilities.

Consider for example a multimedia database: the application may choose to receive a textual description in preference to a reduced quality image (particularly where compression artefacts would be misleading, e.g. in an X-ray photograph). In the same example, suppose that the application was fetching a number of appropriate database records for browsing and selection when the network degraded. In response, the application might instead choose to send some proxy or agent to perform filtering on the records before retrieval over the weaker network link. Finally, if we consider a more advanced application such as an audio-video conference tool the application may choose to adapt by reducing its QoS requirements relating to the individual media streams or by, for example, dropping the video stream in favour of solely audio communications.

In addition to modifying their data transmission patterns applications can also restructure in order to adapt to changes in their communications infrastructure. Consider a groupware application which requires the maintenance of group state. Over a LAN, particularly one which supports multicast such as Ethernet or WaveLAN, group state can be distributed for improved availability and fault tolerance. If members of the group move over to a wider area wireless technology such as GSM, in which group calls and multicast is not currently supported, then concentrating state at a particular point is likely to be simpler to manage. Furthermore, the application may be able to choose to weaken consistency to the mobile group members to improve response and timeliness for better connected members [Cheverst,96].

Supporting adaptation at the application level requires applications that have been specifically written to operate in dynamic environments. However, these techniques also offer the greatest level of adaptability and thus the highest potential gain in performance.

3.4 User Adaptation

The final level of adaptation we will consider is the adaptation of user behaviour. Indirectly, the actions of a system's user affect the majority of the demands upon network resource. If users can be made aware of the implications of their actions, particularly with respect to specific tasks, then many of the problems resulting from network limitations can be

avoided. For example, in an application such as a world wide web browser or a front end to the file transfer protocol if the application provides feedback to the user during a particular operation, perhaps in the form of a progress bar or time until completion monitor, then the user can quickly gain an idea of whether or not they wish to continue with the operation. Thus, through careful design of the user interface to mobile applications, a user can be informed so that they can avoid costly tasks or shift to other, more easily supported ones.

While a significant number of systems have been developed which employ one or more of the adaptation techniques described in sections 3.1 to 3.4, there have been few systems which attempt to employ a range of adaptation techniques at each layer, i.e. system, middleware, application and user. In the following section we describe an experimental system developed at Lancaster as part of the MOST project which has been specifically designed for operation in a dynamic mobile environment and which supports adaptation throughout its architecture.

4. An Experiment in Adaptation

4.1 Background

The MOST project was a collaborative project between Lancaster University and EA Technology which aimed to develop an advanced mobile application to support field workers in the electricity utilities industry. In essence, field workers are responsible for the establishment and maintenance of the electricity distribution network. This work consists primarily of an on-going maintenance and extension programme which ensures the continual improvement of the existing infrastructure and provision for distribution to new property and industry developments. Over and above these routine scheduled procedures are on-demand emergency repairs in response to reported or detected losses of supply. Such repairs must be conducted with the utmost urgency and may range from loss of power to private customers to larger consumers such as hospitals, light and heavy industry. It is important to restore supply to maintain customer acceptability and failure to do so within reasonable timescales typically results in loss of revenue, potentially custom and in many cases penalty payments. The first priority of the field engineer is to restore supply quickly and above all safely and then arrange for repairs as a secondary concern. The network is designed with a certain level of

redundancy to ensure the highest attainable level of supply under fault conditions. Diagnosing a fault, pinpointing and isolating the cause and coordinating the repair requires a wide range of resources and a high degree of coordination and collaboration.

The project begun initially with an extensive requirements capture process and continued to involve end-users through an evolutionary review process which led to the development of a series of concept demonstrator prototypes culminating in the application described in this paper. The involvement with end-users highlighted a number of significant shortcomings within the current working processes which led to many specific requirements and issues to address. These were distilled into the following requirements that needed to be addressed in our work.

Mobility

Field engineers are inherently mobile and require access to both fixed and mobile resources from the field.

Spatially referenced data

Field engineers handle a wide variety of schematic and geographic data, including a variety of maps, engineering drawings and connectivity diagrams. These images are tied to specific regional and geographic coordinates. The application must provide tools for browsing and overlaying these images at specified coordinates.

Multimedia

The map and reference data is contained in both raster and vector form and may also contain hand annotations. It is essential that audio communication is maintained between engineers since data only exchanges are too restrictive given the flexibility of the work and current safety practices.

Collaboration

The application should allow field engineers who are not physically co-located to interact using voice and cooperate using shared maps, schematics and highlighting

tools. Less synchronous information exchange between engineers and other control room staff is also required (i.e. essentially multimedia E-mail).

Interoperability

There is a considerable requirement for interoperability within the utilities. Field engineers must be able to interact with a wide range of legacy systems within their own organisation and frequently with those of neighbouring utilities. In addition, an ever increasing amount of interaction between utilities is required, for instance the Common Street Works Register (CSWR) which aims to coordinate the activities of the many utilities to avoid involuntary disruption and minimise roadworks.

The one significant requirement that was not considered in our system was safety criticality. Field engineers work in an environment in which stale or out of date information is potentially life threatening. Furthermore, it is essential that any actions both verbal and through the system are logged in order that any incidents can be investigated satisfactorily. The safety issue is not considered further in this paper.

The application was developed to run in a test environment consisting of a backbone of workstations linked via Ethernet and portable machines with dial-up cellular links (initially analogue but later GSM). Since an overlay network testbed was unavailable, we developed a network emulator [Davies,95b] which ran over the Ethernet and enabled us to experiment with advanced network scenarios such as the overlay network topology described in section 2.

The following sections consider the prototype architecture in more detail and, specifically, how support for adaptation has been integrated into the system.

4.2 Overall System Architecture

4.2.1 Overview

The application is made up of three components: application code, a supporting distributed systems platform and a dial-up network driver (called S-UDP). The overall architecture is illustrated by figure 1.

The requirement for interoperability identified in section 4.1 is addressed through our use of a standards based distributed systems platform. More specifically, the platform is based on the ANSAware distributed system [Ltd.,92] originally developed as part of the Advanced Networked Systems Architecture (ANSA) project under funding of the Alvey research initiative. We selected ANSAware since the ANSA project and the ANSAware platform have had a profound influence on the development of the ISO/ ITU-T standards for a Reference Model for Open Distributed Processing (RM-ODP) [ISO,95a], [ISO,95b], [ISO,95c]. As a result of this influence, RM-ODP and ANSAware share a number of common features as is apparent from the following description of the ANSAware computational model. A system is described as being comprised of one or more component objects. Each object is made up of encapsulated internal state and one or more interfaces defining how other objects may interact with it. Each interface is defined in terms of a set of named operations which can be invoked. Invocation of an operation takes a set of typed arguments and responds with a termination. A termination consists either of a set of expected typed results or an exception condition. Two forms of invocation are defined, synchronous and asynchronous. In a synchronous invocation the client thread invoking the service blocks for the termination and the client and server are synchronised by the interaction. Asynchronous interactions do not block the client, furthermore, no assurance that the operation has terminated (or even started) is given.

Operational interfaces are obtained through a brokerage service known as the trader (see figure 2). The trader offers a well known interface to objects to enable them to request a reference to a particular service. These interface references may be freely exchanged by objects. An implicit binding to the service is established at first access. The platform features a number of QoS based extensions which are described in more detail in section 4.3.1.

The object oriented model supported by the platform is reflected in the design of the application itself. More specifically, the application has been designed as a toolkit of component modules each of which is responsible for one element of the functionality of the overall application. The main modules and their purposes are described below.

Geographic Information System (GIS) Module

This module provides field engineers with support for viewing and annotating geographical data such as maps and schematic diagrams. The module can work in either stand-alone mode or, with the aid of the conference management module described below, as a real-time synchronous collaboration tool. In the latter case groups of users are able to jointly view and annotate maps in real-time providing similar functionality to shared whiteboards such as [Jacobson,92].

Conference Management Module

This module provides support for establishing and destroying conferences of users. These conferences involve audio communications and collaborative applications such as the GIS module.

Remote Database Access Module

This module supports remote access to a central database which contains customer records.

Structured E-mail/ Job Dispatch

This module supports structured E-mail including annotations and maps produced using the GIS module. The module is primarily designed to be used for job dispatch purposes.

Structuring the application as a toolkit enabled new components to be added simply and facilitated the collaborative development of the application as a whole. Furthermore, each module can be independently controlled promoting configurability for particular tasks and allowing a field engineer to utilise a subset of the available functionality.

Each component module has been implemented as one or more platform objects. The conference management object is responsible for the overall coordination of group activities (such as joining or removing group members and circulating group state delta information). The group conference manager offers an interface to the application modules to enable them to monitor changes in group state and reconfigure themselves appropriately. The group state

in the current prototype is always distributed between the conference participants to maximise fault tolerance and availability (at the expense of the additional communication overhead required by state maintenance).

All interactions between objects (both local and remote) are carried out by platform invocations. In particular, collaboration is coordinated using object invocations rather than, for example, using a windowing system based approach such as X splitting [Abdel-Wahab,91]. Our approach has a number of advantages over windowing system based approaches since it allows views of shared information to be resized independently, rather than constraining all views to exactly the same level of detail, size, resolution and position on each display. This constraint would impair interaction between heterogeneous hardware and software (e.g. field engineers with hand-held PDAs interacting with control room staff who use high performance workstations with large displays) and enforce standardisation on particular windowing system software.

The platform features a number of QoS enhancements which enable the application components to monitor and control interactions with other application components. By incorporating a flow of QoS information throughout the architecture as a whole, the application and supporting platform are able to adapt to changes detected in the underlying infrastructure. The adaptive features and mechanisms are discussed in more detail in the next section.

4.3 Adaptation

As we identified in section 3, applications can make more effective use of resources if they are able to respond to changes in the underlying infrastructure. A wide range of techniques were highlighted which may be applied throughout a system to achieve this. In the MOST prototype, the process of adaptation is facilitated through the provision of a QoS architecture which supports the collation of QoS information with mechanisms to allow the various layers to register their QoS requirements and receive notifications when changes are identified. The QoS support in our architecture is explained in more detail in the following sections.

4.3.1 System

The lowest level of our platform is the Serial-UDP (S-UDP) driver. S-UDP is analogous to SLIP or PPP in that it provides IP packet encapsulation and forwarding over serial lines and dial-up links. S-UDP provides UDP-like point-to-point packet transport, i.e. packets are only delivered to the application layer if they are free from corruption (damaged packets are dropped silently).

In order to support adaptation and to enable us to experiment with new communications APIs S-UDP differs from SLIP and PPP in two significant ways. Firstly, S-UDP provides lightweight support for QoS through an extended interface which allows applications (or more usually the platform) to enquire as to the status of the bearer technology, register for certain QoS events and implement application policies for controlling the agent's dialling mechanism. Secondly, S-UDP incorporates an earliest deadline first scheduler to enable the platform to specify the relative urgency of application invocations.

In more detail, the QoS interface to S-UDP allows applications to register for changes in line state, monitor the absence of contact from a particular host within a time threshold and inform the application (using a backcall address) about changes in the number and earliest deadline of waiting packets (when the line state is down). S-UDP mimics a range of QoS parameters concerning the bearer technology, dial-up and hang-up delays, and cost information (e.g. the charging strategy, so much per unit time or per unit of information etc.). Ideally this information would be made available from the network itself. Currently however, this information is retrieved from a database according to configuration options supplied by the user. Parameters such as the dial-up and hang-up delay are monitored and improve in accuracy over time.

If no dialling policy has been registered with S-UDP (see figure 3), it offers a default strategy which dials immediately upon reception of a packet for transmission. When 30 seconds of idle time have elapsed the connection is dropped. Early work with analogue cell phones which operated per minute billing enabled us to optimise the hang-up control to align

with the boundary of the next charging interval. However, the GSM services bill per second and do not offer the same scope for optimisation.

The QoS callback interface offers an extensible architecture for informing higher layers when significant changes occur, for instance, change of bearer service or dial-up technology. In the current implementation, the S-UDP driver has to be configured manually to switch network technology.

4.3.2 Platform

The distributed systems platform used in the MOST architecture offers a number of additional features over the basic ANSAware platform which have been designed to facilitate adaptation. The most significant of these enhancements is the support for QoS based explicit bindings for both operational interfaces and stream bindings for supporting continuous media. Explicit binding support brings our platform into closer alignment with the RM-ODP standard.

An explicit binding is established between a pair of operational interface references using the following syntax :-

```
! {Handle} <- binder$Bind(ClientIR, ServerIR, QoS)
```

The Bind operation results in a handle to the binding (if successful) and provides a mechanism for checking establishment preconditions according the supplied QoS parameters (the current admission policy simply checks the validity of the supplied interfaces to bind). The handle is a first class interface to a local instantiation of a QoS monitoring object (known as the binding control interface). The binding control interface allows the client/ server to query the status of a range of QoS parameters that are maintained by the platform. In addition, the system supports a callback interface to allow applications to supply bounds of acceptability on many of the QoS parameters and receive callback notifications when bounds are transgressed. This provides the primary mechanism through which adaptation is achieved.

Currently the QoS interface supports the following QoS parameters :-

- Available throughput
- Propagation delay

- Idle time
- Reachability

The first two of these parameters are derived by the platform's RPC mechanism (QEX) by gathering statistics from the round trip time of platform interactions. The statistics are used by the protocol in two ways: firstly, QEX adjusts its retransmission timers to aid it in congestion avoidance and control (the round trip information is filtered using Jacobson's smoothing calculation [Jacobson,88]). Secondly, the information is provided to the QoS monitor to compare against the event registrations made at the binding control interface. The application supplies a window of tolerance for the throughput and delay parameters and will receive a notification if the calculated rate falls outside these bounds. The callback function can specify a variety of actions to take in response to the notification, including renegotiating a different window of tolerance.

The idle time and reachability parameters are valuable additions in a weakly connected environment. Traditional RPC mechanisms often provide mechanisms for generating an exception termination if a server cannot be reached. However, it is often valuable to be able to detect at the server interface whether a client has been able to contact us. For instance, where an application is structured using callbacks or received periodic information updates. The idle time parameter allows the server to specify a maximum time the binding is allowed to remain idle between invocations before a callback notification is generated. The reachability parameter strengthens the assertion offered by the idle time parameter that the absence of communication within the idle time has not been due to communications failure. In essence this assertion maps down to platform level polling (coordination between the platform components ensures that only the most stringent polling requirement is meant and thus attains a greater efficiency than possible at the application level). The reachability parameter allows conventional callback structured applications to be ported to a mobile environment without radical restructuring.

The support for stream bindings in the platform are related to earlier work at Lancaster [Coulson,92]. Stream bindings provide an end-to-end abstraction over continuous media

communication and support arbitrary many-to-many interconnections (*m* sources can be connected to *n* sinks, $m \geq 1$, $n \geq 1$). Each binding is managed by an explicit binding object (called the Stream manager) which accepts either singleton or group interfaces for binding (groups are handled by endpoint group managers). Subject to verification of admission criteria, the QoS of the binding as a whole or individual members of endpoint groups can be adjusted. The QoS parameters include desired throughput, latency and jitter (these are typically set according to the media source, rather than by applications specifically). QoS can be adjusted once the flow has been established through the setQoS() and getQoS() operations. As with operational explicit bindings, explicit stream bindings support a callback mechanism to inform clients of any QoS degradations reported by the underlying transport service (the current implementation uses UDP and thus receives no such notification).

We are currently in the process of adding explicit bindings between operational group interfaces. The revised platform model will enable applications to selectively maintain the notion of group transparency. As with the existing bindings, applications may establish a binding to a group of interfaces and receive a binding control handle. The application may then choose to maintain group transparency, specify a quorum to enable the automatic determination of successful group invocations (reduced transparency) or specify a *message profile* for the group (non-transparent). The message profile enables fine grained control of the temporal, ordering, reliability and cost QoS constraints that is to be placed on specific group members. It is intended that the revised model will facilitate group collaborations that can tolerate weakened consistency, in situations where traditional group semantics would not be viable. The explicit group binding and revised semantics are discussed more comprehensively in [Cheverst,96].

The QoS monitoring, feedback and control exhibited by the platform provides a mechanism through which application adaptation can be achieved. In our current application prototype, component modules can establish explicit bindings with their counterparts on other hosts and keep track of the perceived QoS between hosts. The remote database access tool for example, uses the QoS information to seamlessly adjust the volume of data returned depending on the available throughput (depending on the size of the records and the number

of matches in the found set for a particular query). The application has also been designed to use this information to provide feedback to the user. The next section looks at this facility in more detail.

4.3.3 User

As identified in section 3, in most cases the user is ultimately responsible for the generation of network traffic through their actions and choice of task. If the user's actions can be influenced to choose an appropriate task or mode of working for the available resources, then the network can seem less restrictive, improving perception of the environment and the application's usability.

Figure 4 illustrates the primary interface to the MOST application through which group state is reflected, members are added and removed, and the application's component modules can be controlled. The figure illustrates a group collaboration with two other parties (three including the user whose display we can see), reflected by the photograph icons in the centre. Underneath each user is a column of smaller icons representing the components currently being used by that user. A more complete description of the application and the workings of the interface can be found in [Davies,95a].

The icons under the representation of each user serve two main purposes. Firstly, the icons graphically illustrate the activities being undertaken by each user and may be clicked on with the mouse to toggle whether group operations are propagated to each of the other users. The figure illustrates that the first user is using his GIS module, but the second user is not. The absence of a highlight box around the GIS icon illustrates that group operations will not be propagated to that user. It is thus possible to control within a group conference which of the possible components are being used collaboratively and further, which of the group members are collaborating with that module.

The second and, in the context of adaptation, more important purpose performed by the icons is to reflect the status of underlying communications infrastructure to the user. Each active module establishes an explicit binding to its counterparts on the remote machines. The binding control interface is used to register for QoS events concerning the available

throughput of the channel used for the binding. The notification callback and renegotiation mechanism is used to update the background colour of the icon representing the module. The colour reflects the relative connectivity between the users and is chosen to mimic the signal strength indicator lights traditionally found on radio handsets. Strong green on one users's module against red under another user represents the fact that connectivity to the former user is better than to the latter (in this case offers higher throughput and lower delay). If performance degrades during a particular collaboration, a user would thus be able to gauge which of the users they are collaborating with is causing the bottleneck. To speed up the collaboration, the user may then choose to temporarily suspend the propagation of group operations to the user or users slowing the interaction (ideally with their consent) in favour of sending a state snapshot later on. We assume that audio communication with the other parties is always possible.

The relative connectivity indicators in the group module provide the necessary feedback to allow the user to adjust synchronous interactions to maximise productivity. The application's structured E-mail tool can be used to propagate information that is not time critical or when synchronous collaboration becomes impossible.

Figure 5 illustrates the classic time/space matrix proposed by the CSCW community to describe patterns of collaboration [Rodden,92]. If people meet at the same place at the same time, face to face interaction is possible. If people are at the same place at different times then one can leave a message for the other (perhaps pinned to a notice board). If people interact from different places and different times then one needs to send a message to the other (E-mail). If the people are at different places at the same time however, they can use a conferencing tool (e.g. video conferencing). Conventional systems make a clear distinction between synchronous and asynchronous styles of working. However, we found that in the MOST system that users were able to move seamlessly between synchronous and asynchronous methods of working (or vice versa) by selecting when to propagate information to conference participants. User changes in interaction patterns tended to be carried out in response to the availability of the communications technology as reflected by the user interface. Through the use of a simple colour based system we were able to ensure that the

user required no technical understanding of the communications technologies being used to interpret and respond to the QoS information.

5. Analysis

Throughout this paper we have argued that for the majority of applications effective operation in a dynamic network environment requires adaptation. Furthermore, we have argued that this adaptation can, and indeed should, occur at many levels within the system. We have demonstrated this concept in practice with the development of the MOST mobile collaborative application. During this development process a number of important observations were made.

Firstly, developing adaptation support at multiple levels requires mechanisms for efficient QoS information flow within the system architecture. In common with many platform and protocol architectures, our QoS architecture is layered. QoS information flows from one layer to the next, requiring that each layer translate the information from the layer below before passing it up to the next layer in the architecture (and vice versa). Furthermore, the QoS management mechanism of each layer is similar but disjoint; the QoS management mechanism of S-UDP for example is similar but distinct from that of the platform and so on. This has a number of implications for the system. Firstly, QoS events that are useful to more than one layer have to undergo unnecessary delay and potentially translation by other layers. For example, consider a QoS event reporting that the line state of a particular bearer is now down. This information may be useful to all layers of the architecture, but in the existing architecture will generate several callbacks as the information travels up through the layers. Secondly, since each layer effectively implements its own QoS architecture, the addition of new parameters may require new code in all the layers. Thirdly, there are currently no standards governing how and when QoS information should be passed and by what mechanism. The absence of such standardisation has ensured that a wide variety of entirely proprietary mechanisms have evolved effectively prohibiting interworking between different QoS architectures.

The second observation relates to adaptation at the user level. Supporting adaptation at this level required relatively little work once the supporting infrastructure has been instrumented with the appropriate interfaces. However, adaptation at the user level brings about potential gains in performance in excess of those normally achieved by the system. Clearly adaptation at this level conflicts with the idea of providing mobility transparency. However, since users are normally aware of their own mobility in any case this is, we believe, less of an issue than transparency in fixed network systems.

The final observation we make is that not all QoS information that is valuable to adaptive applications is communications oriented. For example, the battery level may be linked with the power consumption of network devices but is also affected by other peripherals and the CPU itself. A more potent example can be found by considering the problem of selecting which of a range of multiple possible bearer technologies should be used for a particular item of data. The application may wish to gather QoS information from each of the devices in order to discover the best (based presumably on application demands, perhaps in terms of raw throughput, delay characteristics, cost or power consumption). In our current architecture this could only be achieved by first establishing a connection using each of the viable technologies, setting up an explicit binding between application objects on each host and then querying the QoS available to the binding. In this case the close integration of communication and QoS management is a liability.

6. Conclusions and Future Work

This paper has examined the role of adaptation in supporting applications in modern heterogeneous mobile environments. A summary of common adaptive techniques has been presented. These techniques were classified as being at either the system, platform, application or user level. The main body of the paper then described in detail an experiment in developing an adaptive system with support for adaptation at all levels. This experiment was carried out under the auspices of the MOST project and consisted of the development of a collaborative mobile application and associated systems support. In particular, the MOST system including support for multiple network technologies, featured self-tuning protocols

with the ability to propagate QoS information and adaptive application components including a remote database access client and a collaborative GIS tool. Based on this experiment we have been able to make a number of observations which are, we believe, of use to designers of future adaptive systems.

1) *Adaptation support is required at all levels of a system and, consequently, an architecture for propagating QoS information through the system is required.*

2) *Adaptation at the user level can have a significant impact on the overall system design if mobility transparency is not a goal.*

3) *Adaptation support is not concerned solely with communication issues and more integration between communications oriented QoS and other environmental factors is required.*

In our current work we are attempting to address these issues by developing a new platform with integrated support for the full range of QoS parameters, both communications and non-communications oriented. The platform is based on the tuple-space paradigm and makes use of the inherent time and space decoupling present in this asynchronous model to support periods of disconnection. Further details of this work can be found in [Davies,97].

7. References

Abdel-Wahab, H.M. and Feit, M.A. (1991) "XTV: A Framework for Sharing X Window Clients in Remote Synchronous Collaboration," *IEEE Tricomm '91: Communications for Distributed Applications and Systems.*

A.P.M. Ltd. (1992) "An Introduction to ANSAware 4.0" *Architecture Projects Management Ltd., Cambridge, U.K.*, February 1992.

Cáceres, R. and Iftode, L. (1994) "The Effects Of Mobility on Reliable Transport Protocols," *14th International Conference on Distributed Computer Systems (ICDCS)*, Poznan, Poland 12-20.

Cheverst, K., Davies, N., Friday, A. and Blair, G.S. (1996) "Services to Support Consistency in Mobile Collaborative Applications," *3rd International Workshop on Services in Distributed Networked Environments (SDNE)*, Macau, China 27-34.

Coulson, G., Blair, G.S., Davies, N. and Williams, N. (1992) "Extensions to ANSA for Multimedia Computing" *Computer Networks and ISDN Systems* 25: 305-323.

Crane, S. and Dulay, N. (1996) "A Configurable Protocol Architecture for CORBA Environments" *Department of Computing, Imperial College of Science, TEchnology and Medicine*, 1996.

Davies, N., Pink, S. and Blair, G.S. (1994) "Services to Support Distributed Applications in a Mobile Environment," *1st International Workshop on Services in Distributed and Networked Environments (SDNE)*, Prague, Czech Republic.

Davies, N., Blair, G.S., Cheverst, K. and Friday, A. (1995a) "Experiences of Using RM-ODP to Build Advanced Mobile Applications" *Distributed Systems Engineering* 2: 142-151.

Davies, N., Blair, G.S., Cheverst, K. and Friday, A. (1995b) "A Network Emulator To Support the Development of Adaptive Applications," *2nd USENIX Symposium on Mobile and Location-Independent Computing (MLIC)*, Ann Arbor, Michigan, U.S.

Davies, N. (1996) "The Impact of Mobility on Distributed Systems Platforms," *International Conference on Distributed Platforms*, Dresden 18-25.

Davies, N., Wade, S., Friday, A. and Blair, G. (1997) "Limbo: A Tuple Space Based Platform for Adaptive Mobile Applications," *Joint International Conference on Open Distributed Processing and Distributed Platforms (ICODP/ICDP '97)*, Toronto, Canada.

Friday, A. and Davies, N. (1995) "Distributed Systems Support for Mobile Applications," *IEE Symposium on mobile computing and its applications*, Savoy Place, London 95/219: 6/1-6/3.

Friday, A., Blair, G.S., Cheverst, K.W.J. and Davies, N. (1996) "Extensions to ANSAware for advanced mobile applications," *International Conference on Distributed Platforms*, Dresden 29-43.

Hager, R., Klemets, A., Maguire, G.Q., Smith, M.T. and Reichert, F. (93) "MINT - A Mobile Internet Router," *43rd IEEE Vehicular Technologies Conference (VTC)*, Secaucus, New Jersey, U.S.

Hills, A. and Johnson, D.B. (96) "A Wireless Data Network Infrastructure at Carnegie Mellon University" *IEEE Personal Communications* 3: 56-63.

Imielinski, T., Vishwanathan, S. and Badrinath, B.R. (94) "Power Efficient Filtering of Data on the Air," *EDBT*, Cambridge, U.K.

ISO (1995a) ISO/IEC Draft Recommendation X.902, International Standard 10746-1, “ODP Reference Model: Overview”, January 1995.

ISO (1995b) ISO/IEC Recommendation X.902, International Standard 10746-2, “ODP Reference Model: Descriptive Model”, January 1995.

ISO (1995c) ISO/IEC Recommendation X.903, International Standard 10746-3, “ODP Reference Model: Prescriptive Model”, January 1995.

Jacobson, V. (1988) "Congestion Avoidance and Control," *ACM SIGCOMM*, 314-329.

Jacobson, V. (1992) "A Portable, Public Domain Network 'Whiteboard'" April 1992:

Joseph, A., deLespinasse, A., Tauber, J., Gifford, D. and Kaashoek, M.F. (1995) "Rover: A Toolkit for Mobile Information Access," *15th ACM Symposium on Operating System Principles (SOSP)*, Copper Mountain Resort, Colorado, U.S. 29: 156-171.

Joseph, A.D., Tauber, J.A. and Kaashoek, M.F. (1996) "Building Mobile Applications with the Rover Toolkit" *M.I.T. Laboratory for Computer Science*, 1996.

Katz, R.H. (1994) "Adaptation and Mobility in Wireless Information Systems" *IEEE Personal Communications* 1: 6-17.

Katz, R. and Brewer, E. (1996a) "The Case for Wireless Overlay Networks," *SPIE Multimedia and Networking Conference (MMNC)*, San Jose, California, U.S.

Katz, R., Brewer, E., Amir, E., Balakrishnan, H., Fox, A., Gribble, S., Hodes, T., Jiang, D., Nguyen, G., Padmanabhan, V. and Stemm, M. (1996b) "The Bay Area Research Wireless Access Network (BARWAN)," *IEEE COMPCON Spring '96 - 41st International Computer Conference*, Santa Clara, California, U.S.

Kistler, J.J. and Satyanarayanan, M. (1992) "Disconnected Operation in the Coda File System" *ACM Transactions on Computer Systems* 10: 3-25.

Mummert, L.B. and Satyanarayanan, M. (1994) "Large Granularity Cache Coherence in the Coda File System," *USENIX Summer 1994 Conference*, Boston, U.S.

Neuman, B.C., Augart, S.S. and Upasani, S. (1993) "Using Prospero to Support Integrated Location-Independent Computing," *USENIX Symposium on Mobile and Location Independent Computing*, Cambridge, Massachusetts, U.S. 29-34.

van Renesse (1996) "Masking the Overhead of Protocol Layering," *ACM SIGCOMM Conference*, Stanford, U.S.

Rodden, T. (1992) "A Survey of CSCW Systems" *Lancaster University*, 1992.

Schilit, B., Adams, N. and Want, R. (1994) "Context-Aware Computing Applications," *Workshop on Mobile Computing Systems and Applications*, Santa Cruz, CA, U.S.

Schill, A. and Kümmel, S. (1995) "Design and Implementation of a Support Platform for Distributed Mobile Computing" *DSEJ* 2: 128-141.

Waldspurger, C.A. and Weihl, W.E. (1994) "Lottery Scheduling: Flexible Proportional-Share Resource Management," *1st Symposium on Operating Systems Design and Implementation*, Monterey, California, U.S. 1-11.

Wallace, G.K. (1991) "The JPEG Still Picture Compression Standard" *Communications of the ACM* 34: 30-44.

Want, R., Hopper, A., Falcao, V. and Gibbons, J. (1992) "The Active Badge Location System" 10: 91-102.

Zenel, B. and Duchamp, D. (1995) "Intelligent Communication Filtering for Limited Bandwidth Environments," *5th IEEE Workshop on Hot Topics in Operating Systems (HotOS-V)*, Rosario Resort, Orcas Island, Washington, U.S.

Table I - QoS signatures of common networks

Network	**Throughput**	**BER**	**Packet oriented**	**Multicast (group)**	**Dial-up**
Ethernet	10 Mbps	10^{-9}	Yes	Yes	No
RF LANs (e.g. WaveLAN)	242 Kbps - 5.7 Mbps	10^{-8}	Yes	Yes	No
Infra-red	9.6 Kbps - 4 Mbps	$>10^{-6}$	Yes	Yes	No
GSM	9.6 Kbps (corrected)	10^{-8}	No	No	Yes
TETRA	7.2 Kbps-28.8 Kbps (raw, 3 protection levels)	$>10^{-3}$ (raw)	Depends on mode	Yes	Yes

Table II - Classification of common adaptation techniques

Level	Technique	Description
User	Change of working practices	The user can alleviate demands on the network, e.g. change task, swap from synchronous to asynchronous collaboration or specify which tasks are most important to them.
Application	Restructure using agents or delegation of processing	Processing/network intensive tasks can be offloaded to remote sites or pre-processing or filtering applied to remote data (reducing bandwidth requirements and freeing host for other tasks/dozing to save power).
	Use proxy services	The application can use local substitute services based on cached information (often with reduced functionality) while disconnected.
	Change model of interaction	Interactions can be adjusted from polling to event based structures or from RPC to an alternative (perhaps asynchronous) paradigm.
	Reorganise application structure	One example of application restructuring is to change from using distributed state to a centralised architecture to simplify consistency management in unreliable conditions.
	Re-bind to new services	The application may be able to rebind to equivalent services which are easier/cheaper to access. Alternatively, it may be possible to migrate the service or application component.
	Change application demands	New QoS requirements can be negotiated or non-essential bindings dropped. Alternatives may be possible, e.g. lossy encoding.
	Adjust consistency requirements	Groups may be able to tolerate weaker consistency or adjust operations to achieve quorum, yet avoid hard to reach members.
Middleware	On-demand cache management	Information can be fetched only when needed, instead of speculatively, e.g. opening the first page of a document and transferring successive pages later, or retrieving e-mail headers before message bodies.
	Prefetching into the cache	The application can fetch information while the link is good, in case it is required when the link degrades or becomes expensive.
	Apply filtering and compression	The volume of information to transfer can be reduced by compression or filtering non-essential frames from hierarchically encoded data.
	Efficient protocol utilisation of the channel	The transport mechanisms can be adjusted to match channel characteristics, e.g. retransmission/backoff strategies, header compression, error control and handling of asymmetric channels.
Transport and below	Change or introduce new protocols	New protocols can be selected which suit the characteristics of a particular network or appropriate protocols can be introduced (e.g. injecting a reliable data link layer).
	Optimise data for the network	Protocols can adjust their packet sizes to suit different networks. The operating system can adjust the queue sizes onto the network interfaces which impacts on latency, particularly of multimedia streams.
	Optimisation of multicast	Multicasts can be mapped onto the network technology, particularly those with partial or full hardware multicast support.
	Optimise for the characteristics of the network	There are a number of cost and network structure optimisations. For instance, batching data to spread the dialling delays, or transferring additional information while the time is already paid for.
	Reordering of data	The priority or urgency of data may require that it is handled preferentially in scarce bandwidth situations.
	Demultiplexing to multiple networks	If multiple technologies are available simultaneously, it may be advantageous to use several at once.

List of Figure Captions

Figure 1

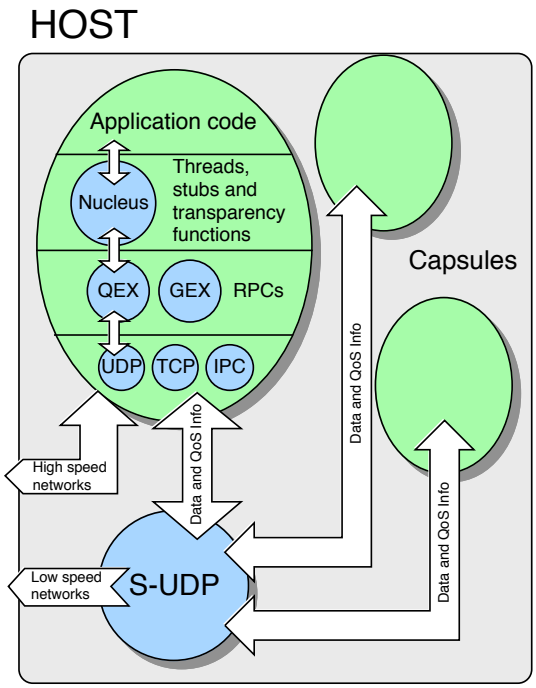

HOST
Application code
Threads, stubs and transparency functions
Nucleus
QEX
GEX
RPCs
UDP
TCP
IPC
Capsules
Data and QoS Info
Data and QoS Info
Data and QoS Info
High speed networks
Low speed networks
S-UDP

Figure 2

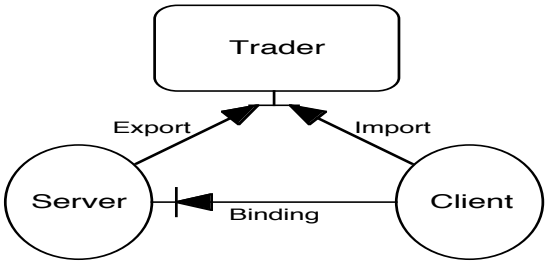


Figure 3

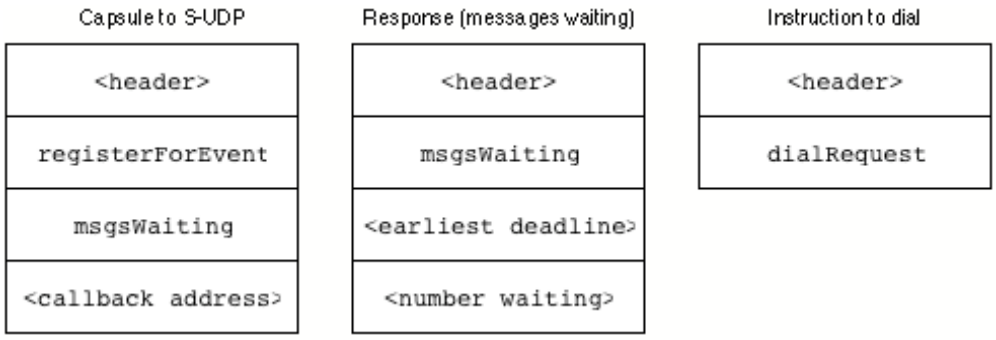

Capsule to S-UDP
<header>
registerForEvent
msgsWaiting
<callback address>
Response (messages waiting)
<header>
msgsWaiting
<earliest deadline>
<number waiting>
Instruction to dial
<header>
dialRequest

Figure 4

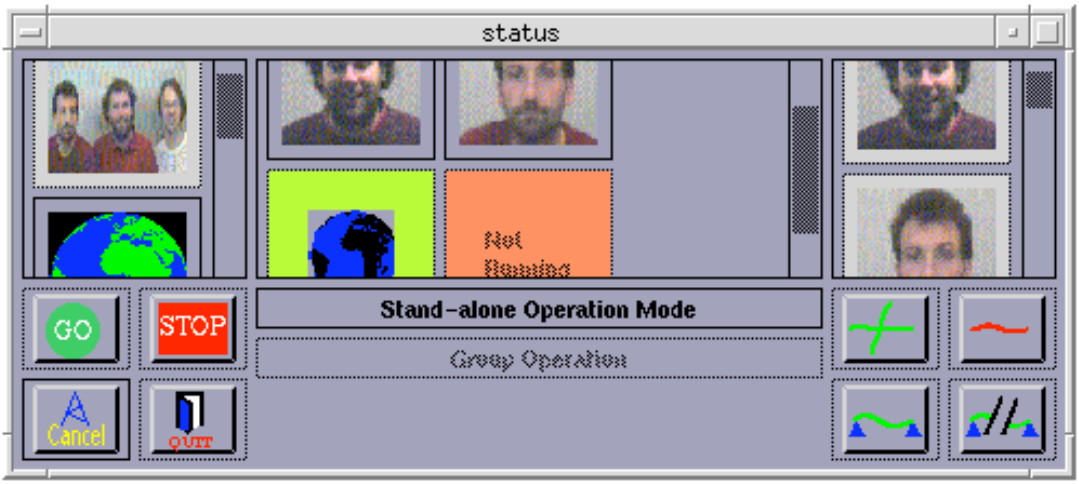
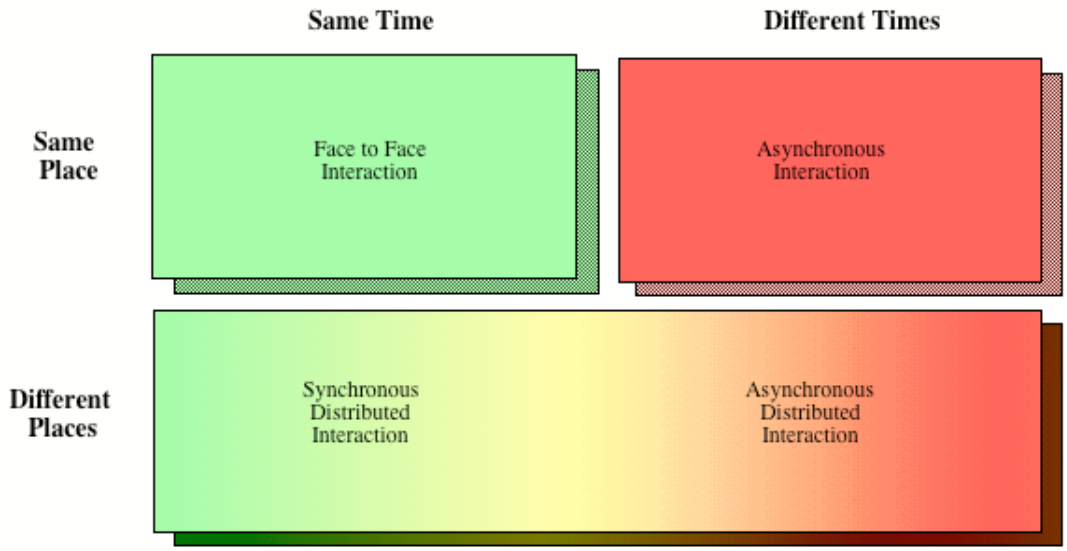

status
Not
Running
GO
STOP
Stand-alone Operation Mode
Group Operation
Cancel
QUIT

Figure 5


Same Time
Different Times
Same Place
Face to Face Interaction
Asynchronous Interaction
Different Places
Synchronous Distributed Interaction
Asynchronous Distributed Interaction